One Block on Top of the Other: Using Minetest to Teach Scrum

Jan-Philipp Steghöfer
jan-philipp.steghofer@gu.se
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden

Håkan Burden
burden@chalmers.se
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden

ABSTRACT

Teaching Scrum using Lego has been an established teaching technique for years. However, the COVID-19 pandemic forced teachers all over the globe to rethink this valuable teaching tool. In this experience report, we show how we transferred our version of a Lego Scrum workshop into the world of Minetest, an open-source variant of Minecraft. We detail our reasoning, the concrete technical and pedagogical challenges, as well as experiences and reflections from the students, us as teachers, our peers, and theoretical frameworks. Finally, we share our materials to enable other teachers to use this new tool.

CCS CONCEPTS

• **Software and its engineering → Agile software development**;
• **Social and professional topics → Adult education**.

KEYWORDS

Scrum, Software Engineering Education, Online Education, Serious Games

ACM Reference Format:
Jan-Philipp Steghöfer and Håkan Burden. 2022. One Block on Top of the Other: Using Minetest to Teach Scrum. In *44nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '22), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3510456.3514157

1 INTRODUCTION

Teaching agile methodologies such as Scrum [25] using serious games has been a popular pedagogical approach ever since the original Lego Scrum exercise was published by Alexander Krivitsky in 2009 [12] and Lynch et al. picked it up for use in university courses [18]. A number of educators have adapted the approach to teach Scrum in different ways in university [23, 26] and high school settings [21]. Lego has since then shown its potential to also teach other software engineering activities (see, e.g., [14, 16]).

However, the COVID-19 pandemic required educators to revise their approaches in the light of restrictions to in-person teaching and a move to online learning. A workshop with many dozen participants in a closed space and in close proximity, as required by the Lego Scrum workshops, was simply no longer possible. Instead, educators moved their teaching to online formats (see, e.g., [22]) that replicate or approximate in-person workshops.

In this experience report, we describe our efforts to replace Lego Scrum workshops with a similar format in which students use Minetest[1], an open source variant of Minecraft to learn the principles of Scrum. The new workshop format follows similar ideas – students use Scrum to build a cityscape or a comparable product with the support of product owners and pre-defined user stories – but relies completely on online interactions between participants.

Our experiences from four workshops in three course iterations show that it is indeed possible to achieve learning outcomes that are very similar to those of Lego Scrum workshops. However, communication and technical hurdles proved to be additional difficulties that are not present in the original.

With this report, we hope to provide a foundation for any educator who wants to teach Scrum using a serious game online. We share our pedagogical concepts, but also many technical details in the hope that others can benefit from them. In addition, we make all used material, including the Minetest worlds, configuration files, and templates for the backlog available to the community[2].

In the following, we first give an overview of how educators are using serious games to teach Scrum in Section 2. We then discuss the educational context of the courses on which we base our experience in Section 3. We describe the pedagogical and technical details of the Minetest Scrum workshops in Section 4 and discuss our experiences and reflections in Section 5 from the teacher, the student, the peer, and the theoretical perspectives. Finally, we provide guidelines and lessons learned in Section 6.

2 TEACHING SCRUM WITH SERIOUS GAMES

There are countless examples of serious games used in software engineering education (see, e.g., [1, 7–10, 19, 20, 24, 29]). At least some of these approaches are used to teach software development processes (e.g., [1, 6, 10]) or, more specifically, Scrum (e.g., [8, 9, 26, 28]). All of these approaches have in common that they attempt to teach a subject that is mostly defined through practical skills and abilities in a setting where students can try out, hone, and apply these abilities directly.

In terms of serious games introducing large-scale agile methodologies, a foundational contribution was set out by Krivitsky [12] who defines a workshop where teams together build a city using LEGO. In the exercise, one team will be responsible for the house with a front garden; another team will build the SUV while a third will deliver the car port. As development proceeds the teams will recognise that the Product Owner wants the house and the car port in the same colour and that the car port has to be big enough to

[1] https://minetest.net/
[2] https://github.com/steghoja/minetest-scrum

ICSE-SEET '22, May 21–29, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9225-9/22/05.
https://doi.org/10.1145/3510456.3514157

fit the SUV. Kropp et al. [13] adapted the workshop, using office materials instead of LEGO, to the decrease the cost of the workshop and introduce shorter build times during sprints. Another take on the exercise is given by Steghöfer et al. [26]. They conclude that the students do manage inter-team collaboration to organise their work but often neglect essential Scrum practices such as estimation and break-down of user stories. Due to time restrictions and students' cognitive load they opted to not address these shortcomings during the LEGO exercise but in a follow-up exercise.

Using LEGO requires physical co-location, something that is not always possible. O'Farrell et al. [22] have a virtual game that reenacts parts of the Scaled Agile Framework (SAFe, [15]). The findings illustrate that digital games can be effectively used to compliment traditional lecture-based approaches to teach large-scale agile development, even if their exercise does not introduce dependencies between teams.

Whether or not innovative learning technologies such as serious games are accepted by students depends on a number of factors [17]. While social influence plays a minor role, positive emotions is more important. The decisive factor is, however, how useful the students perceive a learning activity to be. It is therefore important to align the game to fit the intended learning outcomes. A game by itself will not teach students agile processes and practices; it is how the experiences from playing are tied to the intended learning outcomes that enables that kind of learning.

3 EDUCATIONAL CONTEXT

The Minetest exercise has been used a total of five times in three different course instances, twice in a course on software development methodologies (in Winter 2020, W20) and three times in a course on agile project management (in W20, and Spring 2021, S21). Both courses represent 7.5 ECTS, i.e., five weeks of full-time studies. They are given over ten weeks so that each student is expected to work 20 hours a week, including lectures and reading literature. Both courses previously used Lego Scrum exercises.

3.1 Software Development Methodologies Course

The course is taught to second year bachelor students in a program on Software Engineering and Management. While they have been exposed to agile software development process in previous project courses, they lack the theoretical background to understand and apply Scrum and other approaches correctly. The course addresses these with a number of lectures and exercises and also introduces concepts of software process improvement. In this course, the exercise is used twice: once to allow students to experience Scrum in a controlled manner, and once to allow them to apply an improved version of Scrum that they have created using well-known software process improvement techniques [27]. Both instances of the workshop were attended by around 65 students.

3.2 Agile Project Management Course

The course is taught to different audiences, all on the bachelor level. This includes computer science, software engineering, and industrial economy students in their second and third years of study. The course's focus is on practical application of knowledge and skills. Students are introduced to agile project management with a mixture of lessons and exercises before tackling their project.

The project scope is defined so that the students have to plan, implement and evaluate a project where they do not own the definition of value [4]. The ambition is thus that the students will understand the basic concepts of software engineering and be able to reflect on the relationship between stakeholders, processes and products [5]. The Minetest Scrum exercise is given in the first week of the course to give the students a first experience of Scrum that later can be expanded by lectures introducing the theoretical background for the practices and artefacts.

In the first course instance, we split the 150 participants into two workshops, so that each workshop had 70 students in 15 teams. In the second instance 75 students in 13 teams participated. We aim to have teams of five to seven members, since this is a reasonable size for a Scrum team and results in a total number of teams manageable during supervision and grading.

4 TEACHING SCRUM IN MINETEST

The Scrum workshop as we use it in our teaching is a short and focused activity of no more than four hours in which students go through several Scrum sprints. During this time, students need to work with user stories, discuss and negotiate them with stakeholders, estimate them, plan them, review them with the stakeholders, run retrospectives and make changes and adjustments identified in the retrospectives. That means that students experience all aspects of a Scrum process in a very compressed format.

The Scrum workshop allows students to experience Scrum in a setting that is free from code and allows them to focus on process aspects instead of having to deal with complications arising from setting up complicated toolchains and resolving dependencies. While we acknowledge that using a set of technologies as described below re-introduces some of these complications, the barrier of entry is still significantly lower. The only new piece of technology students need to contend with is Minetest. All other tools are already used in our teaching and students are familiar with them.

Students are asked to create either a city or a castle out of the materials available in the game world. They are provided with a product backlog on Trello. Teaching assistants act as product owners, and teachers act as Scrum coaches and observers.

In terms of intended learning outcomes, the workshop addresses the following aspects, inspired by [26]:

- Introduce Scrum in a way that focuses on the process, rather than on the product.
- Inspire students to take initiative, create and follow plans, and take responsibility for their work.
- Expose students to the need to understand resource limitations and plan accordingly.
- Sensitivise students for the need of iterative retrospectives and improvements.
- Allow students to interact with a stakeholder whose ideas and wishes might not be congruent with their own.

In the following, we go through the different aspects of the workshop and how we set them up. We will start by giving a high-level overview of the workshop in Section 4.1. Then, we will briefly discuss Minetest itself (Section 4.2). Following that, we will consider

the selection of the world in which the workshop takes place in Section 4.3 and communication channels in Section 4.4.

4.1 Overview of the Workshop

The Scrum workshop is scheduled for approximately $3\frac{1}{2}$ hours. A rough breakdown is shown in Table 1. The workshop is facilitated by two teachers who act as observers, Scrum coaches, and referees. Three or four teaching assistants take on the roles of the product owners. These TAs have prepared for the workshop by sorting the pre-defined user stories in the product backlog by priority and assigning epics and stories to themselves. This means that each story has a clearly denoted, responsible product owner.

Students get a priori information about the technology used in the workshop and are asked to install and familiarise themselves with Minetest at least a week before the workshop takes place. On the day of the workshop, the teacher opens the server and sends out the URL and password to all students. At this stage, all students can enter the game, but they do not have the ability to interact with the environment. This means that the world remains pristine until the building phase starts. All students join a Zoom meeting with pre-defined breakout rooms for each group and for each PO. When the workshop starts, student groups have already been formed.

Introduction to the Workshop. The workshop begins with an introduction by the teachers. This includes a "product vision" (a screenshot of an actual cityscape of castle in Minecraft). The teachers also go through the basic structure and ceremonies of Scrum, discuss the roles of Scrum Master and Product Owner, and mention the importance of the Definition of Done. In our courses, these elements have been introduced before, so a brief repetition and a chance to ask questions is all that is needed. The students then move into the breakout rooms for their groups. They are asked to:

- decide who will act as Scrum Master;
- come up with an initial Definition of Done (which should be applicable to **all** teams); and
- perform an initial effort estimation of the user stories they would like to take.

The students do not have a lot of time to perform those tasks – in fact, we do not expect them to estimate the effort of more than one or two user stories. Students usually fail to realise that the Definition of Done (DoD) should not be team-specific; as a matter of fact, it is not rare that a part of the students still assumes that each group will build their own cityscape or castle.

Once the time is up, we call students back into the main Zoom meeting. We hold a very brief discussion focused on the DoD and the need to revise and reflect on estimations. Students also get the chance to ask final questions, but experience shows that at this stage, students are usually eager to start.

The First Sprint. When the first sprint begins, students leave for their breakout rooms again and the teacher activates the "interact" privilege on the Minetest server which means that all students are now able to manipulate their environment.

Students often skip the *sprint planning*. In most cases, students have not thought about how to "claim" a user story for their team or how to negotiate with other teams who takes which story. In many cases, this leads to some buildings being built multiple times.

Table 1: Rough breakdown of the Minetest Scrum workshop – in an actual workshop, concrete timing varies depending on factors such as observed issues and perceived fatigue.

T - 7 days	Send information about Minetest to students
T - 1 day	TAs sort product backlog by priority and assign user stories
	Set up Minetest server and test it
T - 3 hours	Send URL and password of Minetest server to students
T - 0	Introduce the workshop and the students' tasks in joint Zoom meeting
T + 20 minutes	Students join breakout rooms with tasks: select Scrum master, discuss backlog items, estimate a few user stories, discuss DoD
T + 45 minutes	Brief discussion of the DoD and the estimations
T + 50 minutes	First sprint starts
T + 65 minutes	First sprint review
T + 75 minutes	Joint sprint retrospectives with observations from the teachers
T + 80 minutes	Team sprint retrospective
T + 85 minutes	Second sprint starts
	Go through three or four sprints; breaks in between are highly recommended.
T + 180 minutes	Final sprint retrospectives start
T + 190 minutes	Workshop conclusion: brief summary by teachers, establish lessons learned and give students chance to observe their creations in "fly" mode
T + 205 minutes	End the workshop

Students also realise how few resources are available – depending on the world, either how little wood or how little space. Experience shows that their initial estimations usually do not account for the resource collection. Teams also realise that it is inefficient for the entire team to gather resources or build. Engagement with the product owners is low; instead, students interpret the (incomplete) user stories any way they see fit. These initial difficulties are very similar to what has been observed in Lego workshops [26].

The first sprint thus usually does not result in many complete buildings. The tight schedule, the issues with resource gathering and the chaos in the teams mean that most stories are far from completed. We revoke the "interact" privilege for all students and call all teams to a *joint review* in the Zoom meeting and ask the product owner of each team to present the accomplished work. One of the teachers with "fly" privileges shares the screen during that time so that all students have the same view. Issues such as duplicate buildings or incorrect assumptions are usually determined at that stage. We then provide some feedback on the way of working in a *joint retrospective*. In particular, we point out the fallacy of assuming too much about resources and requirements, urge students to understand the exercise as an effort of the entire class to produce *one* cityscape or castle, and emphasise that coordination within the teams, between teams, and with the product owners is necessary.

The teams are then sent back to the breakout rooms to do a *team sprint retrospective* with a strict time limit. It is usually during that time that teams start coordinating their efforts in Slack. At times,

Scrum Masters meet in the joint Zoom meeting or request that we create a specific breakout room for a Scrum of Scrums.

Subsequent Sprints. With the main hurdles out of the way, students tend to become more productive in later sprints. They start engaging each other in coordinating who takes which user story. They request a way to make the assignments visible in the Product Backlog (cf. Section 6). They also become more adept at collecting resources and splitting up the team into gatherers and builders.

A common issue, however, remains communication with the Product Owners. Students have a hard time identifying what they do not know about a user story and asking Product Owners the right questions. We encourage PO to react with absurd requests to open questions, e.g., to respond to "What material do you want the house to be?" with "Gold". This way, we try to teach students to treat the discussion about user stories as negotiations in which they need to convey what is feasible to the POs.

To save time, we keep the main part of the sprint reviews private. The teams are encouraged to join the breakout rooms of the POs during review time and show the increment to the POs there. We as the teachers participate in selected reviews only as observers and use the joint retrospective which is part of every sprint to bring up issues that we see. Of course, each team continues to conduct their individual retrospectives after the joint ones. Students also sometimes elect to hold an additional program retrospective attended by the Scrum Masters.

We repeat the sprints three or four times, depending on scheduling and how much time is used up by the joint retrospective and discussions. We also try to gauge the mood of the students and their motivation, a difficult task in a purely online setting. It is also important to plan for breaks in between sprints – we often break after the first sprint and after the third.

Sign-off and Follow-up. We dedicate the final 15 minutes of the workshop to a summary. We describe to the students what they have experienced once more and relate their challenges to what developers experience when working on software in the industry. It is important at this point to tell students that their experiences were not unique but that the time pressure, the poor user stories, and the overall sense of being ill prepared for the task at hand all were intentional to drive the point home that working in an agile manner does not mean working in a chaotic manner. We also reassure students that the shared experience they have just had will be useful in the rest of the course.

During subsequent lectures, labs, and other interactions with the students, we then bring up this shared experience repeatedly. We remind students about the effects of assumptions, missed communication, and how they continuously improved their way of working based on their reflections in the sprint retrospectives.

4.2 Minetest

Minetest is an open source version of Minecraft that is available for many operating systems and system architectures[3]. It is also available via the package managers of popular Linux distributions, albeit not always in the current version. Its availability and the relatively easy installation lowers the barrier of entry for the students. Minetest has modest system requirements and even machines that are several years old are able to run it at acceptable speeds.

The basic principle of the game is the same as Minecraft: the player is represented by an avatar in a game world that consists of blocks of different materials. It is possible to *mine* these blocks, i.e., pick them up, store them in an inventory, and place them elsewhere. Depending on the material of the blocks, tools are necessary to mine them. These tools can be created by *crafting* where different blocks are combined into a block of a new material or to a piece of equipment. With crafting, the player cannot only create tools, but also other implements such as windows, doors, or torches. More complicated implements such as windows require specific materials as well as specialised tools such as a forge. Therefore, multiple steps need to be completed to arrive at the final product.

The game world is limited in size. Depending on the world in which the player finds themselves, different materials are readily available or need to be painstakingly found. Especially ores are often hidden in mines that are accessible from the surface. Players need to make sure that they do not get stuck in these mines.

Minetest can be started as a dedicated server to which players connect. The server administrator has the ability to configure the server in detail and provide rules and constraints for the players.

Performance Considerations. To use Minetest as an environment for the Scrum Workshop, we ran it as a standalone server on a 2019 Macbook Air (1,6 GHz Dual-Core Intel Core i5, 16 GB RAM). It was connected to the internet via a 100 MBit/s fiber connection at one of the teacher's homes. The server was accessible from the internet via Dynamic DNS. Both the hardware and the network connection proved to be sufficient to run the game world for about 100 students. The main limiting factor was the CPU power: the server used less than 1 GB of RAM and the internet connection was used to about 60%. However, students sometimes experienced lag, especially when logging into the game. Both CPU cores were fully used at all times. Some experiments with the server settings for drawing distance and maximum number of packets sent per player per time step did not yield any significant performance improvements.

Security. Since the server is only online for a limited period of time, security is not a major concern. However, some precautions should be taken, as always when exposing a computer to the internet and running any publicly accessible service on it.

In terms of the network setup, it is important to ensure that only the port the Minetest server runs on is exposed to the internet, especially when running the server on a home machine exposed via Dynamic DNS. A firewall on the router should protect the home network against prying eyes and all other services that might pose security threats should be disabled during the workshop.

Minetest itself allows to password protect the server, a feature that should certainly be used. Avoid passwords with special characters at this increases the chance that students mistype them. An important thing to note is that the password is identical for all users. That means that, in theory, anyone can login with the credentials of the teachers. To avoid this, we recommend to set the server password to one value, ask all teachers and TAs to log in using this password, and then set the server password to the value for the students. Teachers and TAs can then log in with the original passwords and students cannot take over those accounts. This is

[3] https://www.minetest.net/downloads/

especially important since Minetest uses the username to assign privileges (see below).

The `whitelist` mod (see below) allows accepting only predefined users. While this is a way to alleviate the risk of having players other than students on the server, this was not an issue for us and we did not want to take on the additional administrative overhead.

Finally, it is advisable to disable that the server is advertised in the global server list. This can be easily achieved by setting `server_announce = false` in `minetest.conf`.

Configuring Mods and Permissions for the Scrum Workshop. Mods are plugins that extend the functionality of the Minetest game engine. We are using a number of mods in the workshop, mostly to improve control over the privileges of the students and to allow us to teleport them around, e.g., when they have gotten stuck in mines underground or cannot find their way to their starting pyramid. The mods we are using are listed in Table 2.

Most importantly, the `edutest` mod offers two important pieces of functionality: it allows to change the `interact` privileges for all students and it allows to teleport students to the current location of the teacher's avatar. The first functionality is useful to ensure that students only manipulate the game world during the sprints. Experience shows that students do not focus on the reviews and retrospectives when they still have the ability to collect and build. This also ensures that the breaks are observed. The latter functionality is helpful if students get stuck and cannot free themselves any longer. As a side effect, `edutest` also allows to grant `fly` privileges to all students which is a nice touch at the end of the workshop to let everyone admire the final product.

In order for `edutest` to work as expected, it is important that the teachers have the `teacher` privilege (automatically giving them access to the UI and the chat commands) and students have the `student` privilege. The latter is easily accomplished using the entry `default_privs = shout,student,fast` in `mintest.conf`.

We recommend granting `teacher` or any other privileges such as `fly` to during the initial login to establish the special password for teachers and TAs as described above. During that time, the server host can grant privileges to individual users. These privileges are persisted and survive a server restart.

4.3 Finding a World for the Scrum Workshop

Finding a fitting world for the Scrum workshops was a surprisingly difficult task. While there are ways to manually create worlds for Minetest[4], this possibility was dismissed since this task is too time consuming. Instead, we relied on the randomized generation of worlds that Minetest offers. Even though this process was still labor intensive, it is possible to evaluate generated worlds relatively quickly. Overall, we went through about 120 worlds before settling on two worlds with different properties that we used in the workshops. An early decision was to limit the map size to 240 tiles square. This was to restrict the participants of the workshop to an area that provides enough space, but is small enough to traverse within less than a minute.

After reviewing a couple of worlds, we settled on a number of criteria:

- There needs to be exposed land (i.e., the land should not be fully submerged).
- There should be sufficient flat land to serve as a building surface.
- Sufficient building material that can be mined without tools (e.g., wood, dirt, sand) should be exposed.
- There should be some diversity in the landscape, e.g., there should be mountains or lakes.

The two worlds we settled on (cf. Fig. 1) adhere to these characteristics but are very different. While our standard world offers a relatively flat central valley with steep mountains around it and almost no vegetation, our alternative world is almost fully covered with trees and lakes, but is still relatively flat and offers sufficient ground to build on once trees are cleared away.

These two worlds pose very different challenges: while the standard world offers plenty of space but very scarce wood resources (which are needed to craft tools such as pick axes which in turn are required to harvest stone and other materials), the alternative world has plenty of wood but very little space without clearing a significant proportion of the forest. Both cases allow for a teachable moment: if students are not aware of these constraints, how can they make accurate estimations of their user stories? If students assume that wood will be available or that space will be available, they will underestimate the effort required to acquire these resources. As a secondary effect, this also allows us to point out to the students that they make assumptions (in that case about which resources are available) that are not always tenable. This effect is especially pronounced if two workshops take place after each other as in the Software Development Methodologies course. The students just assume that they will get the same world with the same issues – and are then very surprised to find quite different challenges.

Spawning and Navigation. When students enter the game, they spawn in the world at random points. However, since Scrum is a team-based effort, they need to find each other. For this purpose, we have placed pyramids throughout the world and distribute a map to the students before the workshop that shows where each team should meet (cf. Fig. 2). The map was created with the open source tool Minetest Mapper[5]. While some students still struggle to find their bearings, especially with limited sight distances, locating the right pyramid is usually not an issue that takes up more than a couple of minutes at the beginning of the workshop. Still, students should have the time to find their starting point. Interestingly, once the starting pyramid is located, students tend to also build right beside it. It is rare that they dismantle it for building material.

4.4 Communication Channels

One of the major challenges of moving the Scrum workshop online is communication. While the Lego Scrum workshops we used to conduct had all students and POs within earshot, we now need to rely on technology. The broadcast of a message has become significantly more difficult as getting a feeling for whether such a broadcast was actually received. In the online Scrum workshop, there is no equivalent of standing on a chair and shouting.

[4]Via the WorldEdit mod: https://github.com/Uberi/Minetest-WorldEdit

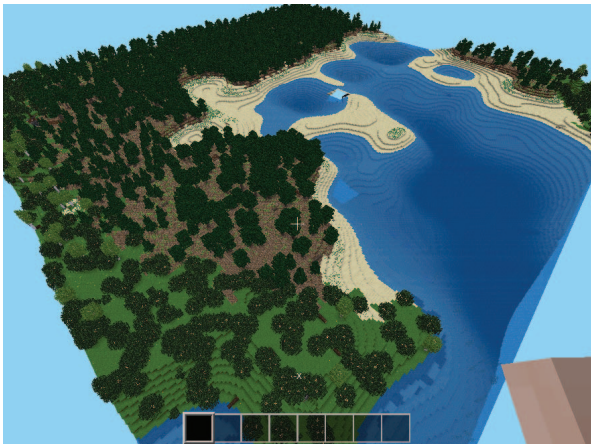
[5]https://github.com/minetest/minetestmapper/; command line tool with Windows binaries, can be compiled on Linux and macOS. GUI with binaries for Windows and Linux available at https://github.com/adrido/MinetestMapperGUI-Bundle

Table 2: Mods used to facilitate the Minetest Scrum workshop

Mod	Description
`edutest`[1]	Provides a UI for teachers to limit student privileges en masse and some useful tools such as teleporting students to the teacher's avatar's current location.
`edutest_chatcommands`[2]	Provides chat commands to access functionality of `edutest`. (Optional)
`unified_inventory`[3]	Gives all players easy access to crafting recipes and other helpful functionality. For teachers, it also provides access to the UI of the `edutest` mod.
`whitelist`[4]	Allows to only accept players with specific user names. Can be used as a means to address security concerns. (Optional)
`worldedit`[5]	Allows editing the game world. Used primarily to position the starting pyramids in the world.

[1] https://github.com/zeuner/edutest [2] https://github.com/zeuner/edutest-chatcommands
[3] https://github.com/minetest-mods/unified_inventory [4] https://content.minetest.net/packages/Zughy/whitelist/
[5] https://github.com/Uberi/Minetest-WorldEdit

(a) The standard world we used for some of the workshops. Students were asked to build a cityscape.



(b) The alternative world we used for one workshop. Students were asked to build a castle.

Figure 1: Two different worlds used in the Scrum workshops.

Instead, we use Slack and Zoom to facilitate communication. Slack is helpful for anything that can be consumed asynchronously: reminders, additional information, and messages to individual students. Zoom is the broadcast format for when we need everyone's attention: general announcement, the teachers' reflections, and part of the sprint reviews take place here. In addition, we use Trello for backlog management. A tool that we do not use is the Minetest chat: while Minetest has a built in chat function, messages appear and disappear quickly and are very easy to miss.

Slack and Zoom. To facilitate the workshop, we use Zoom, the video conferencing tool provided by our university for use in teaching. We gather all students in Zoom at the beginning of the workshop where we introduce the setting and the tasks. Each team has a pre-assigned breakout room which makes it easy to split the teams up and have them work on a specific assignment and then call them back for discussions and further announcements. The students are used to working this way and the technical infrastructure is stable and reliable. One disadvantage is that messages delivered via the "Broadcast" function are often overlooked since they only appear on the screen briefly.

Instead, we use Slack. The students usually create a Slack workspace (or, equivalently, a Discord) which we use to make public service announcement and to get in touch with individual groups. We do also use Zoom's ability to jump into the breakout rooms, but if we want to inform students about something without disturbing their flow, Slack is a better option. It also allows us to communicate directly with individual students, if the need arises, and gives students a way to contact us without leaving their breakout rooms. Slack is also very useful for inter-team communication, both in the stage where the students need to come up with a joint Definition of Done, but also to coordinate dependencies between user stories.

In order to facilitate communication with the Product Owners, we also have individual breakout rooms for them. This way, students have easy access to them and can easily find them to ask questions. In our initial workshop, we encouraged the POs to wander from team to team, but this caused frustration with the students since they had a hard time locating the POs or needed to hop into another teams breakout rooms to ask their questions.

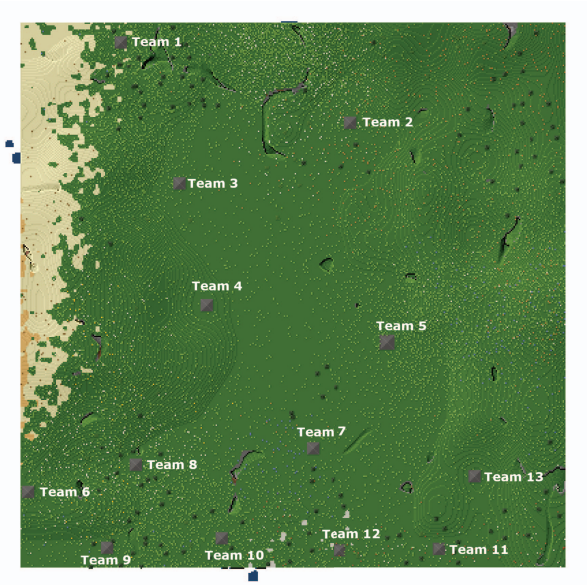


Figure 2: A map of the standard world with the teams' starting points denoted as pyramids.

Backlog Management in Trello. The product backlog is stored as a Trello board. As mentioned above, by default, only teachers and TAs can make changes to the board while students can only view it. This is a precaution to prevent vandalism and to ensure that students cannot interfere unintentionally with each other on the board. At the same time, this is also a disadvantage: students do not feel like they are in control of the board and can use it actively to structure their work. Since they need to go to a PO for any change they want to make, delay is built in. Once students realise how important the backlog is for coordinating with other teams (usually after the first sprint), they tend to accept this limitation, however, and work with it. Alternatively, we also consider giving the Scrum Masters write access to the board (cf. Section 6).

The backlog serves as a semaphore in the sense that teams mark which stories they have taken, thus communicating to other teams that a story is no longer available and to the PO which team is working on it. They very rarely request changes to the user stories, though, be it in terms of how they are formulated or to add new acceptance criteria. However, this might be an effect of the lack of direct access. We certainly perceive more engagement when teams work with physical backlogs in the Lego version of the workshop. In any case, POs move stories between the columns of the backlog when requested and refer to the backlog during the reviews.

We also ask the teams to record the Definition of Done they agreed on in the backlog. Unfortuanately, the DoD is not an artefact that students actively work with during the workshop. With everything else that is going on during the workshop, there is little time to update the usually very lackluster DoD that has been created as part of the initial preparations. Having said that, we do encourage POs to ask about DoD during reviews and for students to reconsider the initial version in our joint retrospectives. This is rarely done in a concerted effort with the involvement of all teams and rather often performed by a single team that thus imposes an updated DoD on all without prior consultation.

5 EXPERIENCES AND REFLECTIONS

This section details prominent experiences in relation to the workshop, both from a student and a teacher perspective. By relating these to prior research and educational frameworks the outcome is in line with Brookfield's four lenses where an educational intervention is assessed from the students', the teachers' and the peers' point of view as well as the theoretical perspective [2]. By reflecting on these experiences and perspectives we have come up with a list of guidelines for what we think are important aspects to consider when conducting a Minetest or similar exercise. The guidelines are not bound to a single perspective or lens, rather derived from combinations of experiences and aspects, so we have chosen to present them in the following section.

5.1 Student Perspective

We are using two datasources to reflect on the exercise from a student perspective. In doing so, we acknowledge that these datasources have not been intended to be used to evaluate the efficacy of the workshop. However, they are both reflections by the students about how the workshop impacted their learning. In the case of the course evaluation reports, they explicitly contain feedback students provided on the workshop or the course as a whole. The datasources we use are:

- Course evaluation reports from the Agile Project Management course in W20 and S21 and the Software Development Methodologies course in W20. These evaluation reports are created by the university based on a standard set of questions administered as an online questionnaire. Participation in the questionnaire is not mandatory and the response rates are correspondingly low. However, experience shows that in particular students that are unhappy with the course do respond. There were no specific questions about the Minetest workshops, but many students mentioned it in the freetext answers.
- Reflection reports from the Software Development Methodologies Course taught in W20. Students were asked to describe their experiences in both workshops and how they applied Scrum in it. These reports acted as the examination in the course and did not focus on Minetest, but rather the application of the process.

Feedback from Course Evaluations. The course evaluations show that students generally appreciate the Minetest workshops. One of the freetext questions in the questionnaire is "Which aspect of the course should be kept for next year?". Six out of eleven students that answered the questionnaire for Agile Project Management S21 mentioned the workshops here. Likewise, eight out of 20 students that answered the question for Software Development Methodologies W20 mentioned keeping the workshop. Only four students provided more detailed responses which we discuss in the following. Interestingly, the Minetest workshops were not mentioned at all in the course evaluation of Agile Project Management W20.

A positive feedback that shows that the workshop goals are achieved was provided by one student:

> "The Minetest exercise was great. It was fun, it taught us about using Scrum and dividing up into sprints and was also a great team-building exercise."
>
> (Agile Project Management S21)

However, of the few students that provided longer responses, three also mentioned issues with the workshop, in particular the relative lack of instructions provided:

> "Difficult to apply what we learned during the lectures in the workshops since the workshops were very stressful. Students should be given more instructions to understand the purpose of the workshop better"
>
> (Software Development Methodologies W20)

> "The Minecraft scrum intro had a bit vague instructions. Out group didn't really understand that resources was limited until after the first sprint for example"
>
> (Agile Project Management S21)

In addition, the difficulty to access the teaching assistants who were the designated product owners were also mentioned explicitly:

> "The minetest-thing was very fun and greatly appreciated by our group. Two things about it: We were confused on what we were doing in the beginning, maybe be more precise in the information. Also, there were long waits for talking to the TA's, so long that we did not get the time to do so in the designated time space."
>
> (Agile Project Management S21)

Overall, this feedback from the reflection reports tells us that we are on the right path. When it comes to the criticism that limited information is available, we fully acknowledge that we provide minimal instructions to students. However, this is done on purpose: first of all, this allows us to demonstrate which impact implicit assumptions have on the way of working; second of all, the workshop is complex enough as it is, and providing additional instructions would increase the cognitive load. Since students run through several iterations, they also have the opportunity to reflect on what happens and make changes in subsequent iterations, therefore being able to adapt to their environment. This is also an important aspect of the workshop (cf. Section 4).

Learning from Reflection Reports. When it comes to the concrete learnings students take away from the workshops, the reflection reports from the Software Development Methodologies W20 course show that the intented learning outcomes are largely achieved.

In terms of creating and following a plan, it is clear that most students underestimate the need for such a plan. Students go into the workshop without a concrete idea of their tasks, how to distribute their work, or how to achieve the desired outcome. Even though creating a plan for the workshop was a specific task in the course, these plans are not concrete enough and do not take the environment in which the workshop takes place into account. Students assume that a basic Scrum approach with incremental delivery will be enough to be successful.

A major issue arises due to the rather vague user stories. The pedagogical reason behind this vagueness is to get students to discuss requirements with the Product Owners to refine them and make sure that the stakeholder wishes are clarified. However, this often does not happen and students instead make up requirements themselves:

> "Concern about the available information in user stories was risen in the group as a lot of the process revolved around expectations of their content . With large tasks and few specified requirements, assumptions had to made on the fly. Leaving room for interpretation proved to have a negative impact on delivery speed."
>
> (Software Development Methodologies W20)

However, students learn from this confusion during the workshop and become better once the POs reject their increments in the first sprint review. In the final sprint, students are usually very good at eliciting requirements from the POs, making proposals and asking questions helpijg them narrow the scope (instead of open questions as "how many windows do you want?" which often lead to considerable feature creep).

Indeed, the workshop is one of the few experiences in the student's curriculum where they are asked to coordinate across teams. This takes many students by surprise:

> "[M]y team had not given much consideration to the program-level planning, assuming that we would have no control over it and so we had not tried to co-ordinate with the other teams before the workshop, which may have been both possible and also extremely beneficial"
>
> (Software Development Methodolodogies W20)

In order to achieve such coordination, communication is essential. However, students found the sheer number of communication channels confusing, as also mentioned in Section 4.4:

> "During the Scrum workshop it was not clear which communication channel should be used to contact stakeholders and other teams. Which communication channel to use was never explicitly defined in the process used."
>
> (Software Development Methodolodogies W20)

Apart from these challenges, which are similar to the ones reported for Lego Scrum exercises [26], there were also some Minetest-specific challenges to note. One in particular was that students were unfamiliar with some of the finer points of the game such as crafting and sometimes had trouble navigating the game's environment:

> "In some situations, we were not able to find the required materials and forge tools like pickaxe to mine some resources, which wastes time during the shortened sprints. [...] [T]here was wrong information on how to make glass and team members were not aware of some functionality such as the "/killme" function to utilize when trapped into the deep underground."
>
> (Software Development Methodolodogies W20)

This issue is somewhat surprising as students had been made aware well in advance what to expect. A lack of preparation on the part of the students can never be fully avoided, though. As one student put it: "other members taught me how to perform the basic functions on Minetest" which shows that it is possible to use a limited amount of time to transfer the crucial skills.

5.2 Teacher Perspective

Our teacher perspective is derived from the notes we took during and after the workshops. Being intimately familiar with Lego Scrum workshops of which we conducted several dozens over the years, we focused our observations mainly on differences and if the Minetest workshops replicate the learning and teaching experience of the Lego versions.

The high-level observation is that, *yes, the Minetest Scrum workshops are a good stand-in for the Lego Scrum workshops*. The five times we have executed them, they fulfilled the exact same function as the Lego versions. In particular, comparing the reflection reports from Software Development Methodolodogies in Winter 2019 to those in Winter 2020, students struggled with the same issues (preparedness, requirements, communication, etc.) as described above. Qualitatively, we thus replicate the experience of the in-person Lego Scrum workshops.

When it comes to the issues of using Minetest specifically, we believe that the platform provides a relatively low hurdle. Installation, gameplay, and connecting to the gameserver are very simple. The system is stable enough for the purposes of the workshop and can be run on commodity hardware with a common home internet connection. Even though crafting and other advanced topics in Minetest are not familiar to all students, they are significantly easier to pick up then programming concepts and not essential to participate in the workshop successfully. Therefore, the format we use is still significantly more accessible than a similar format in which code would be produced.

One drawback of Minetest compared to LEGO is that the world, and therefore the integration area, is much bigger. It takes time to navigate between increments and not all students are prepared to follow someone around. That makes the review more cumbersome and less instructional compared to the reviews conducted with LEGO. In the latter case the integration area consists of two or three tables and all students can gather around, the review takes place in one place and all are present. We have tried to mitigate the drawback with Minetest by not focusing on all increments together, but parallelise the reviews so that each PO conducts there own review. This can cause teams to move between reviews if they work towards several POs during the same sprint. Another option is to let it take time and prepare accordingly in terms of breaks.

5.3 The Peer Perspective

When it comes to the differences between running an in-class exercise versus a virtual one, O'Farrell et al. have only positive experiences [22]. Their digital game can be used both as a stand-alone learning environment or as a complement to the in-class material. They found that it was easier to schedule a virtual game in relation to the restrictions of the participants, that the virtual game benefited from having less theoretical material and that the virtual game gave more opportunities for hands-on experiences. During the restrictions on physical meetings due to the COVID-19 pandemic, the game fostered a sense of presence among the participants that was seen as engaging and a good way to get to learn new colleagues.

O'Farrell et al. also point to the practical nature of their exercise and how it supplements the more traditional and theoretical approaches to introducing large-scale agile in an educational context [22]. They suggest to include dependencies between teams to better prepare them for what large-scale agile offers in terms of complexity. This claim is made more explicit by Kropp et al. who conclude that their game for learning large-scale agile facilitates learning through personal experience, social learning and construction of values and value identity [13]. By personal experience they mean that the students can use the exercise for "appropriation of experiential knowledge accomplished by working in a social or production environment". This is done through a process where the participants learn by relating themselves and their own effort to the work done across roles and teams, including conflict resolution and community approval. Finally, value identity is accomplished as the students actively participate in discussions about what is value for whom and why a specific increment is valuable. We find that the Minetest workshop likewise creates ample opportunities for practical experiences that allow for social learning and discussions on value creation.

The shared experiences are something that both students and teachers can relate back to. This is helpful during the lectures when the more theoretical points are discussed as they can be related to actual situations or artefacts from the exercise. The shared experience can also be useful later on during supervision as it is sometimes easier to understand what the students have gone through if there was a similar experience during the exercise. This resonates with findings reported by Steghöfer et al. [5, 26] who used LEGO instead of a digital game for achieving the same outcome.

5.4 Theoretical Frameworks

We have chosen the concept of cognitive apprenticeship [3] to theorise on the experiences of the exercise. The idea is that we can reason around learning cognitive skills in the same way as manual skills were taught in traditional apprenticeships. This is done through the interaction between the ones who know, the masters, and the ones who want to know, the apprentices. The transfer of knowledge is enabled by six different methods – modeling, coaching, scaffolding, articulation, reflection and exploration.

In terms of modeling there are ample opportunities for the masters to reason around how they came up with an initial scrum board or why the resources are limited and in this way show their own way of learning and thinking about Scrum concepts and practices. Coaching happens when the masters give advice and feedback on how the teams have performed their sprints or used the available artefacts supporting the development, such as how they have (not) updated the scrum board. Scaffolding is when the masters set up an environment that is safe for the apprentices to try out their new competences and the whole exercise can be seen as a scaffolded environment. Articulation can occur during the retrospectives when the apprentices describe their own reasoning and how it came that they chose to split responsibilities in a certain way or skip estimation because it seemed difficult. Reflection is what happens when they then describe what the outcome of their actions were and what they would like to change for the next sprint or their following coding project. The coding project is also an opportunity to further explore the new competences and how they can be used.

The sprint breakdown into planning, doing and reflecting mirrors Kolb's phases in the learning cycle – plan, experience, reflect and conceptualise [11]. Keeping this in mind helps us tailor what we want the students to experience during the exercise to create a shared basis for future discussions on agile concepts and practices.

6 GUIDELINES AND LESSONS LEARNED

By reflecting on the experiences presented in the previous sections we compiled a number of guidelines and lessons learned to help educators create a rewarding and educational virtual exercise.

Joint Sprint Reviews. Getting all students together for a joint sprint review is very difficult. First of all, reviews are time consuming. Going through all user stories tackled by all teams takes a lot of time and is not very interesting for most of the participants. Second of all, students do not have the same view of the game world and it is hard for them to see where a certain building is located.

We have addressed this by having short joint sprint reviews of no more than five minutes where we only look at one or two user stories. During this time, students are not able to interact with the game world. We have called them back from the breakout rooms in Zoom into the common meeting. One of the teachers shares their screen and a randomly selected group presents their work and the PO gives feedback. We couple this with a joint sprint retrospective where we provide some general feedback, e.g., on quality aspects and on the way the team worked.

After the short joint reviews, teams join the breakout room of the PO responsible for their user stories. This at least parallelises reviews into a number of tracks defined by the number of POs. We found that four POs for about 80 students is a reasonable number. POs are told to give short, directed feedback and move to the next group as quickly as possible. Once a team is finished with their reviews, they join their own breakout room for their retrospective.

Parallel communication channels. Many different communication channels with overlapping functionality make it difficult to ensure that everyone is up to speed. There are at least three different chats at the students disposal: Minetest, Zoom, and Slack. This in turn creates a cognitive load for some students which is worth keeping an eye on. On the one hand, it is important to have a concrete discussion about the impact of handling multiple new concepts and activities in an agile project; on the other hand, it is also important that the students have the opportunity to create a manageable working space and keep a sustainable pace throughout the exercise and the course.

Handling stress. Keeping this sustainable pace also requires the students' stress level. Gauging this is difficult in an online setting. During a Lego workshop, it is easy to sense the energy in the room and how stressed the students are just from the way they act and talk to each other. With opportunities for observation limited, teachers need to make up for this in an online workshop. We tend to join different breakout rooms to get a feeling for the mood in the teams. We also regularly ask the teaching assistants acting as Product Owners about their impression.

To mitigate stress, we introduce breaks at appropriate times, but also bring up the factors that cause the stress. All of the issues mentioned above (communication, resource management, planning) contribute to stress. By pointing out the causes and making them visible in the joint sprint retrospectives, we give students a chance to reflect on their stressor and address them. In general, we observe that stress levels go down significantly in the later sprints.

Backlog management. An important aspect of Scrum is that the team takes charge of the sprint backlog. Since we suggest to host the product backlog in Trello and share it between groups, this is not easy to accomplish. In particular, we do not want to give all students permission to manipulate the board since this could wreak havoc simply by a lack of coordination. However, students should have access to backlog to indicate that they selected user stories or move them through the columns on the sprint backlog.

We have tried to accomplish this by asking students to go through the POs and ask them to make the updates. This turned out to be too cumbersome and incurred too much of a delay, causing students to abandon the practice and, essentially, the backlog as a whole. An option that worked better was to give the Scrum Masters access to the backlog. This worked much better, but requires some additional administration. This can be set up before the workshop, however, at least when the Scrum Masters are known in advance.

Students are not well-prepared. When the exercise starts, we see a lot of issues that stem from a lack of preparedness. Students do not have the game installed, have not tested it, or are not familiar with controls. Some students have prior experience of specific settings for platforms similar to Minetest and expect the same features to be available for them during the exercise.

It is worthwhile to have a dedicated TA for tech support in the beginning. A practical tip is to be aware of expectations, both among teaching staff and students, and have a plan for handling them. It can, e.g., be reasonable to let students understand the hard way that there are limited resources and no ability to fly if that creates a learning opportunity for discussing resource management and competences. If it only creates frustration it might be worth clarifying this at the outset. Again, the game is an opportunity for learning experiences and as a teacher you should tune the exercise to fit your learning objectives.

Accessibility. It is a clear disadvantage of the Minetest workshop that it is unsuitable for students with certain impediments such as blindness. While it is possible to include blind students in the Lego exercise as they interact with physical artifacts, Minetest is not accessible in the same way. We do not have a good solution for this issue, but would like to make sure that teachers are aware that they might not be able to include everyone in the class.

7 CONCLUSION

In this experience report, we describe how Minetest, an open-source alternative to Minecraft can be used as an effective tool to teach the use of Scrum. Based on our experience with Lego Scrum workshops, we propose a structure for the workshop and provide technical and pedagogical guidance for the successful execution of the workshop.

We strongly believe that this teaching tool is valuable independent of the COVID-19 pandemic. In many situations, online teaching is applied on a regular basis. Being able to hold workshops in online settings to teach software development processes is therefore a valuable ability even when we are not hiding in our bedrooms.

REFERENCES

[1] Ufuk Aydan, Murat Yilmaz, Paul M Clarke, and Rory V O'Connor. 2017. Teaching ISO/IEC 12207 software lifecycle processes: A serious game approach. *Computer Standards & Interfaces* 54 (2017), 129–138.

[2] Stephen Brookfield. 1998. Critically reflective practice. *Journal of Continuing Education in the Health Professions* 18, 4 (1998), 197–205.

[3] J. S. Brown, A. Collins, and P. Duguid. 1989. Situated cognition and the culture of learning. *Educational Researcher* 18 (1989), 32–42.

[4] Håkan Burden, Jan-Philipp Steghöfer, and Oskar Hagvall Svensson. 2019. Facilitating entrepreneurial experiences through a software engineering project course. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, Montreal, Canada, 28–37.

[5] Håkan Burden and Jan-Philipp Steghöfer. 2019. *Teaching and Fostering Reflection in Software Engineering Project Courses*. Springer, Singapore, 231–262.

[6] Alejandro Calderón, Manuel Trinidad, Mercedes Ruiz, and Rory V O'Connor. 2019. An Experience of Use a Serious Game for Teaching Software Process Improvement. In *European Conference on Software Process Improvement*. Springer, 249–259.

[7] Rafael Oliveira Chaves, Christiane Gresse von Wangenheim, Julio Cezar Costa Furtado, Sandro Ronaldo Bezerra Oliveira, Alex Santos, and Eloi Luiz Favero. 2015. Experimental evaluation of a serious game for teaching software process modeling. *ieee Transactions on Education* 58, 4 (2015), 289–296.

[8] Adler Diniz De Souza, Rodrigo Duarte Seabra, Juliano Marinho Ribeiro, and Lucas E da S Rodrigues. 2017. SCRUMI: a board serious virtual game for teaching the SCRUM framework. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 319–321.

[9] João M Fernandes and Sónia M Sousa. 2010. Playscrum-a card game to learn the scrum agile method. In *2010 Second International Conference on Games and Virtual Worlds for Serious Applications*. IEEE, 52–59.

[10] Ivan Garcia, Carla Pacheco, Andres Leon, and Jose A Calvo-Manzano. 2020. A serious game for teaching the fundamentals of ISO/IEC/IEEE 29148 systems and software engineering–Lifecycle processes–Requirements engineering at undergraduate level. *Computer Standards & Interfaces* 67 (2020), 103377.

[11] D. A. Kolb. 2014. *Experiential learning: Experience as the source of learning and development* (2nd ed ed.). FT Press.

[12] Alexey Krivitsky. 2011. Scrum Simulation with Lego Bricks. https://www.hacerlobien.net/lego/Otr-005-Scrum-Simulation.pdf

[13] Martin Kropp, Andreas Meier, Magdalena Mateescu, and Carmen Zahn. 2014. Teaching and learning agile collaboration. In *2014 IEEE 27th conference on software engineering education and training (CSEE&T)*. IEEE, 139–148.

[14] Stan Kurkovsky and Stephanie Ludi. 2018. LEGO-based Active Learning Exercises for Teaching Software Development. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 1052–1052.

[15] Dean Leffingwell. 2018. *SAFe 4.5 reference guide: scaled agile framework for lean enterprises*. Addison-Wesley Professional.

[16] M.W. Lew, T.B. Horton, and M.S. Sherriff. 2010. Using LEGO MINDSTORMS NXT and LEJOS in an Advanced Software Engineering Course. In *IEEE 23rd Conference on Software Engineering Education and Training (CSEE&T)*. 121–128. https://doi.org/10.1109/CSEET.2010.31

[17] Fernando Rodríguez López, Mario Arias-Oliva, Jorge Pelegrín-Borondo, and Luz María Marín-Vinuesa. 2021. Serious games in management education: An acceptance analysis. *The International Journal of Management Education* 19, 3 (2021), 100517.

[18] Thomas D Lynch, Michael Herold, Joe Bolinger, Shweta Deshpande, Thomas Bihari, Jayashree Ramanathan, and Rajiv Ramnath. 2011. An agile boot camp: Using a LEGO® -based active game to ground agile development principles. In *Frontiers in Education*. IEEE, F1H–1.

[19] Beatriz Marín, Matías Vera, and Giovanni Giachetti. 2019. An Adventure Serious Game for Teaching Effort Estimation in Software Engineering.. In *International Workshop on Software Measurement and International Conference on Software Process and Product Measurement (IWSM-Mensura)*. CEUR Workshop Proceedings, 71–86.

[20] Michael A Miljanovic and Jeremy S Bradbury. 2016. Robot on! A serious game for improving programming comprehension. In *Proceedings of the 5th International Workshop on Games and Software Engineering*. ACM, New York, NY, USA, 33–36.

[21] Marcello Missiroli, Daniel Russo, and Paolo Ciancarini. 2016. Learning agile software development in high school: an investigation. In *Proceedings of the 38th International Conference on Software Engineering Companion*. IEEE, 293–302.

[22] Emer O'Farrell, Murat Yilmaz, Ulas Gulec, and Paul Clarke. 2021. PlaySAFe: Results from a Virtual Reality Study Using Digital Game-Based Learning for SAFe Agile Software Development. In *Systems, Software and Services Process Improvement*, Murat Yilmaz, Paul Clarke, Richard Messnarz, and Michael Reiner (Eds.). Springer International Publishing, Cham, 695–707.

[23] Maria Paasivaara, Ville Heikkilä, Casper Lassenius, and Towo Toivola. 2014. Teaching Students Scrum Using LEGO Blocks. In *Companion Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE Companion 2014)*. ACM, New York, NY, USA, 382–391. https://doi.org/10.1145/2591062.2591169

[24] Selene Ramírez-Rosales, Sodel Vázquez-Reyes, Juan Luis Villa-Cisneros, and María De León-Sigg. 2016. A serious game to promote object oriented programming and software engineering basic concepts learning. In *2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT)*. IEEE, Puebla, Mexico, 97–103.

[25] Ken Schwaber. 1997. Scrum development process. In *Business Object Design and Implementation*. Springer, 117–134.

[26] Jan-Philipp Steghöfer, Håkan Burden, Hiva Alahyari, and Dominik Haneberg. 2017. No silver brick: Opportunities and limitations of teaching Scrum with Lego workshops. *J. Syst. Softw.* 131 (2017), 230–247. https://doi.org/10.1016/j.jss.2017.06.019

[27] Jan-Philipp Steghöfer. 2018. Providing a baseline in software process improvement education with lego scrum simulations. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 126–135.

[28] Christiane Gresse von Wangenheim, Rafael Savi, and Adriano Ferreti Borgatto. 2013. SCRUMIA—An educational game for teaching Scrum in computing courses. *Journal of Systems and Software* 86, 10 (2013), 2675 – 2687. https://doi.org/10.1016/j.jss.2013.05.030

[29] Michalis Xenos and Vasiliki Velli. 2018. A serious game for introducing software engineering ethics to university students. In *International Conference on Interactive Collaborative Learning*. Springer, Cham, 579–588.